Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

филиал «Минский радиотехнический колледж»

Допущен к защите

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( А.С. Козел)

**КОМПЬЮТЕРНАЯ ИГРА «СЛИЯНИЕ ЧИСЕЛ»**

Пояснительная записка

курсового проекта по предмету

«Основы алгоритмизации и программирования»

МРК КП 2-400101 34 091ПЗ

Выполнил учащийся группы 1к9393

/ Гуринович Р.Д./

2023

«Минский радиотехнический колледж»

# КОМПЬЮТЕРНОЙ ИГРА «МОРСКОЙ БОЙ»

Пояснительная записка

к курсовому проекту по дисциплине

«Основы алгоритмизации и программирования»

КП0К.939325.106.081 ПЗ

Руководитель / А. С. Козел /

Учащаяся гр. 0К9393 / М. М. Кичигина /

2022

1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Содержание**  Введение 3   1. Постановка задачи 4    1. Описание предметной области 4    2. Сравнительный анализ существующих решений 5    3. Информационная база задачи 11    4. Функциональное назначение 12 2. Проектирование задачи 14    1. Алгоритм решения задачи 14    2. Логическое моделирование 14    3. Выбор и обоснование инструментов разработки 18 3. Программная реализация 23    1. Физическая структура 23    2. Описание разработанных модулей 24 4. Тестирование 28 5. Применение 30    1. Руководство пользователя 30   Заключение 34  Список использованной литературы 35  ПРИЛОЖЕНИЕ А 36 | | | | | | | | | | |
|  |  |  |  |  |  | | | | | |
|  |  |  |  |  |
| Изм*.* | Лист | № докум. | Подпись | Дата |
| Разраб. | | Гуринович Р.Д. |  |  | Компьютерная игра  «СЛИЯНИЕ ЧИСЕЛ»  Пояснительная записка | Лит*.* | | | Лист | Листов |
| Провер. | | Козел А.С. |  |  |  |  |  | 2 | 55 |
| Реценз. | |  |  |  | МРК | | | | |
| Н. Контр. | |  |  |  |
| Утверд. | |  |  |  |

2

# Введение

**Введение:**

Игра «Слияние чисел» является примером игры с простым геймплеем и интересным исполнением, которое смогло завоевать огромную популярность среди пользователей. Основная суть данной игры заключается в перемещении плиток с числами на игровом поле, игроку необходимо объединять плитки в одну с более высоким числом до тех пор, пока не будет получена плитка с числом 2048 или пока на поле не закончатся возможные ходы.

Цель данной курсовой работы заключается в создании игры «Слияние чисел» на языке программирования C++ с использованием библиотеки SFML. В процессе работы будут рассмотрены следующие вопросы: алгоритмы и структуры данных, применяемые в игре, методы реализации игровых механик, работа с различными библиотеками языка программирования C++, для реализации графического интерфейса игры будет использована библиотека SFML.

# Постановка задачи

* 1. **Описание предметной области**

Компьютерные игры – это сложные программы, которые служат человеку для обучения или развлечения и рассчитаны на разные возрастные группы. Существует большое количество разнообразных жанров игр, наиболее популярными являются жанры: ролевые игры, симуляторы, логические игры, аркады, стратегии.

Игра «Слияние чисел» относится к жанру логических игр и направлена на развитие логического мышления. Цель игры - перемещать и объединять плитки с числами на игровом поле, чтобы получить плитку с числом 2048.

Игровое поле представлено в виде сетки плиток, которые могут быть объединены и перемещены. Каждая плитка имеет свое число, которое удваивается при объединении с другой плиткой.

Одним из интересных аспектов игры «Слияние чисел» является возможность получать более высокие числа, чем «Слияние чисел», что делает игру еще более захватывающей и вызывающей.

Игра начинается с двух плиток, случайно расположенных на игровом поле, плитки содержат в себе числа 2 или 4. Игроку представляется возможность перемещать плитки в любом направлении - вверх, вниз, вправо или влево - при помощи стрелочек на клавиатуре. При каждом движении на свободной плитке появляется новая плитка с числом 2 или 4, случайно выбираемым игрой.

Если при перемещении плиток оказывается ситуация, что в одну сторону сдвигаются плитки с одинаковым значением, они объединяются в одну плитку, у которой число будет равно сумме чисел двух объединенных плиток. Например, если две плитки с числами 2 объединить, то получится новая плитка с числом 4. Если плитки с числами 4 объединить, то получится новая плитка с числом 8 и так далее.

чем меньше игровое поле, тем больше шанс совершить несколько ошибок так, что на игровом поле не будет возможности слияния плиток и больше не будет свободного места для появления новых плиток.

Победа в данной игре определяется ситуацией, когда на одной из плиток значение становится равным установленному, в большинстве случаев данное значение равно 2048, однако оно может изменятся в зависимости от режима игры.

Победа - условие, при котором один из игроков потопил все корабли противника и выигрывает игру.

# Обзор существующих аналогов

Сейчас в игровой индустрии предоставлено бесчисленное разнообразие игр, вследствие этого все новые идеи представляют собой комбинацию старых. Поэтому для разработки лидирующего на рынке, приятного и удобного программного средства для пользователя, необходимо изучить существующие популярные решения и выделить основные недостатки и преимущества, определить ключевые особенности тенденций в данном направлении. Таким образом, нужно проанализировать аналоги.

«2048 for Windows» - классическая игра, реализованная в виде программы для ПК. Имеет простой и удобный интерфейс, представляет выбор размера игрового поля, победного значения. Интерфейс данной игры представлен на рисунке 1.

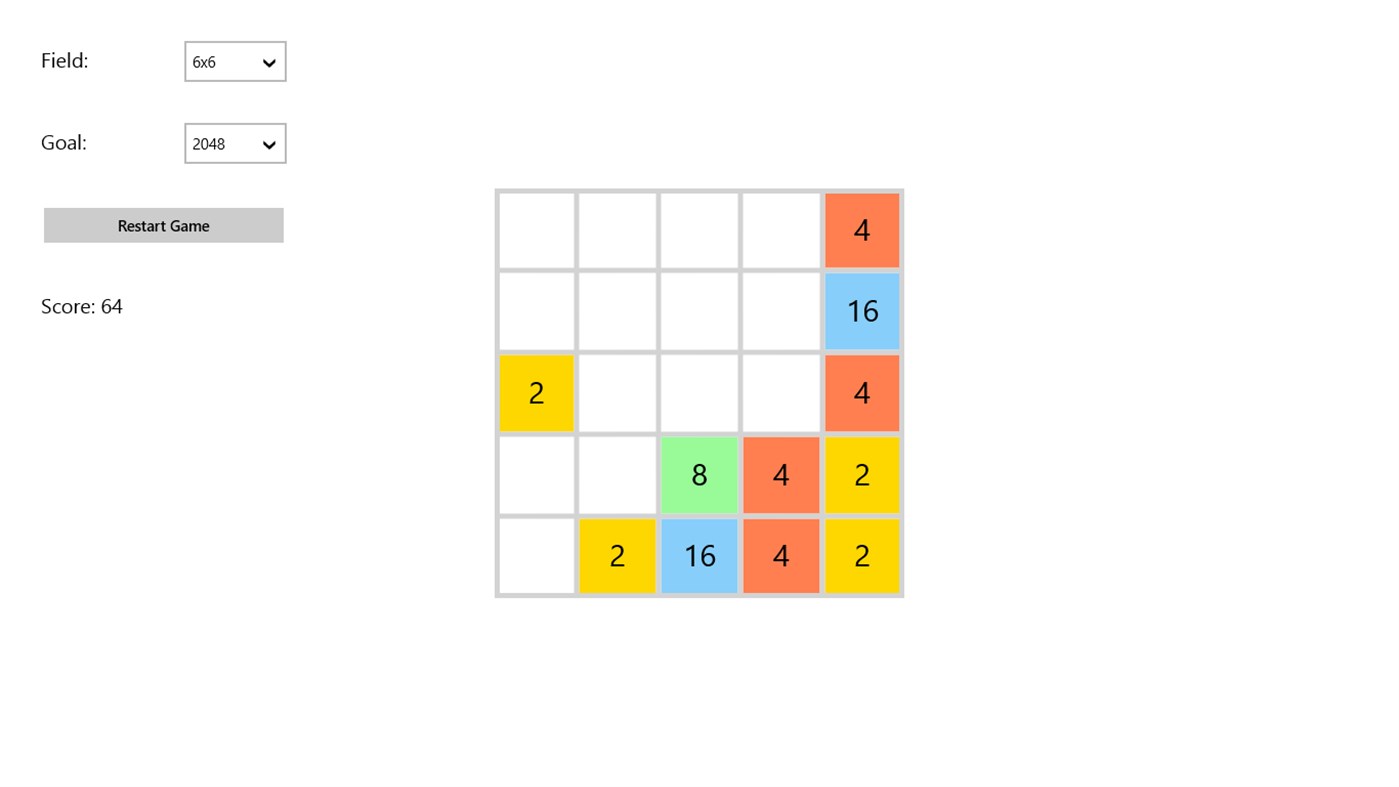


Рисунок 1 – Интерфейс игры «2048 for Windows»

Данная игра имеет следующие положительные стороны: выбор размера поля, выбор числа, до получения которого продолжается игровой процесс. Также можно заметить, что все плитки на игровом поле, имеют разную расцветку, в зависимости от их номинала, имеется отображение набранных игроком очков.

Недостатки данной программы заключаются в невозможности сохранения наилучшего результата.

«Игра 512 1024 2048» - игра, которая поддерживается большим количеством операционных систем. На рисунке 2 представлен интерфейс данной программы.

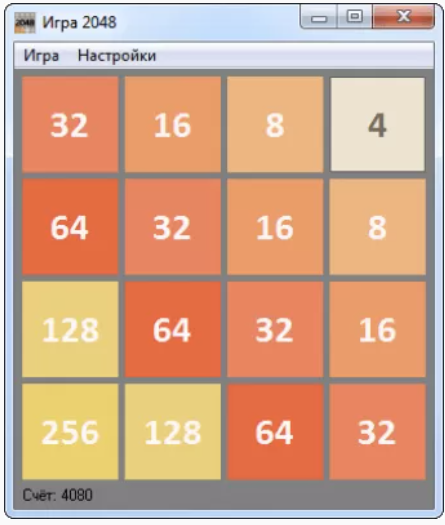


Рисунок 2 – Интерфейс игры «Игра 512 1024 2048»

Положительные стороны данной игры: данная игра имеет настройки игрового процесса, возможность выбора победного номинала, отображение текущего результата во время игрового процесса.

Недостатками данной игры также является невозможность сохранить наилучший результат, также в данной игре нет возможности изменения размера игрового поля.

«2048 Game Professional for Windows». Интерфейс данной игры представлен на рисунке 3.

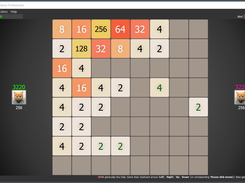


Рисунок 3 – Интерфейс игры «2048 Game Professional for Windows»

Данная игра имеет следующие положительные черты: предоставление выбора различных режимов игры, вариативность выбора размера поля и победного значения, также в ней реализована возможность изменение темы интерфейса игры, реализовано сохранение наилучших результатов.

В качестве недостатков данной игры можно отметить большое количество различной рекламы, а также нет возможности изменения языка интерфейса на русский язык.

Данный жанр игры ограничивает изменение игрового сюжета, однако в играх могут быть дополнительные режимы, возможность соревноваться с другими людьми по сети.

Большое количество аналогов подтверждает тот факт, что игра «2048» пользуется большой популярностью и спросом среди пользователей.

# Информационная база задачи

Информационная база – совокупность данных, систематизированных по определенным признакам и используемых для решения различных задач.

Информационная база (ИБ) ─ это определенным способом организованная совокупность данных, хранимых в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач.

Входные параметры рабочего потока ─ это элементы данных, используемые для выполнения функции. Выходные параметры рабочего потока ─ это элементы данных, которые получаются в результате выполнения действия.

Для создания логической игры используются следующие входные данные: игровое окно, а также текстуры для игрового поля и интерфейса игры.

Выходные данные будут включать полученный результат, количество очков, которое набрал игрок за время игры. Очки начисляются игроку во время игрового процесса за слияние плиток с одинаковым числовым значением.

Основными компонентами игры являются игровое поле, плитки с числовым значением и игровой процесс. Игровой процесс будет реализовываться за счет алгоритмов проверки соседних элементов, а также на перемещении элементов двумерного вектора.

Написание данного курсового проекта является хорошей практикой применения существующих навыков программирования и их развития. Помогает глубоко разобраться в функционале графических библиотек языка C++, а также приобрести необходимый опыт разработки компьютерных игр.

Реализация игры «Слияние чисел» на языке программирования C++ с использованием библиотеки SFML может быть хорошей задачей для студентов, которые хотят изучить язык программирования C++ и научиться использовать графические библиотеки для создания игр. Такой проект может помочь студентам и начинающим разработчикам улучшить свои навыки программирования, практически применить знания и приобрести опыт разработки компьютерных игр.

Таблица 1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Кнопка «Начать игру» | Запуск игрового процесса |
| Кнопка «Закончить игру» | Завершение игрового процесса |
| Кнопка «Настройки» | Выбор кораблей и лимита времени |
| Кнопка «Сохранить» | Сохраняет изменения |
| Кнопка «Начать заново» | Сбросить расположения всех кораблей на вашем и вражеском поле |

# Функциональное назначение

* Функциональным назначением игры «Слияние чисел» на языке программирования C++ с использованием библиотеки SFML является:
* Игра «Слияние чисел» развивает различные стороны человека, такие как логическое мышление, аналитические способности, так как игрок анализирует текущую ситуацию на игровом поле, также развивается концентрация и внимания игрока.
* Игра «Слияние чисел» также развивает стратегическое мышление, так как игроку необходимо продумывать свои действия на несколько шагов вперед, чтобы добиться наилучшего результата.
* Игра Слияние чисел подходит для широкого диапазона возрастов и пользователей. Она может быть интересной и полезной для детей и взрослых. Дети могут использовать эту игру для развития своих умственных способностей. Взрослые также могут наслаждаться игрой, чтобы тренировать свой ум и провести свободное время с пользой.

Для реализации игрового процесса будут реализованы след. функции

# Проектирование задачи

* 1. **Алгоритм решения задачи**

Алгоритм – четкая последовательность действий, выполнение которой дает какой-то заранее известный результат

После изучения данных сведений разработчик может приступить к разработке приложения, где требуется составление намного более подробного алгоритма решения задачи:

* разработка окна основного меню. Визуализация дизайна окна, продумывание функционала кнопок и написание программы его работы;
* более подробный разбор каждой кнопки меню. Написание подробного кода, который максимально описывает её возможности и саму программу в целом;
* разработка программы для самого окна игры. Добавление в него уже своих кнопок, которые помогают ходу решения задачи;
* окончательная установка связей между всеми окнами игры.

# Логическое моделирование

* Диаграмма деятельности (Activity Diagram) - это графическое представление последовательности действий и поведения системы или процесса. Она используется для моделирования и визуализации потока управления или деятельности, отображая различные состояния, действия и переходы между ними.
* Цели диаграммы деятельности:
* Понять логику и последовательность выполнения процесса или операции.
* Идентифицировать возможные проблемы, узкие места или улучшения в процессе.
* Согласовать представление о деятельности или процессе между заинтересованными сторонами.
* Основные элементы диаграммы деятельности:
* Действия (Actions): Представляют собой конкретные шаги или операции, которые выполняются в процессе.
* Решения (Decisions): Позволяют ветвиться потоку управления, основываясь на определенных условиях.
* Параллельность (Parallelism): Показывает параллельное выполнение нескольких действий или процессов.
* Соединители (Connectors): Указывают направление и последовательность потока управления.
* Начало и конец (Start and End): Обозначают начало и завершение диаграммы.
* Диаграмма вариантов использования (Use Case Diagram) - это графическое представление функциональных требований к системе, иллюстрирующее, как различные актеры (пользователи или внешние системы) взаимодействуют с системой для достижения определенных целей.
* Цели диаграммы вариантов использования:
* Понять функциональные требования к системе.
* Определить взаимодействие между актерами и системой.
* Согласовать представление о функциональности системы между заинтересованными сторонами.
* Основные элементы диаграммы вариантов использования:
* Актеры (Actors): представляют роли, которые взаимодействуют с системой. Актеры могут быть пользователями, другими системами или внешними компонентами.
* Варианты использования (Use Cases): Описывают конкретные функциональные возможности или сценарии использования системы. Каждый вариант использования описывает, как система и акторы взаимодействуют для достижения определенной цели.
* Отношения (Relationships): Указывают связи и взаимодействия между акторами и вариантами использования. Наиболее распространенными отношениями являются отношение включения (include) и обобщения (extend).

Упомянуть приложения

Компонентами диаграммы деятельности являются:

− переходы – отношение между двумя состояниями, показывающее, что объект, находящийся в первом состоянии, должен выполнить определенные действия и перейти во второе состояние, как только произойдет указанное событие и будут удовлетворены указанные условия;

− дорожки – часть области диаграммы деятельности для отображения деятельностей, за которые отвечает конкретный объект;

− объекты – экземпляры класса, которые уникально идентифицируются значениями атрибутов, определяющими их состояние в данный момент времени.

Диаграмма деятельности программного средства компьютерная игра

«Морской бой» представлена в графической части КП0К.939325.106.002ПЛ.

# Выбор и обоснование инструментов разработки

В качестве языка программирования для создания игры используется язык программирования C++, основной характеристикой которого является возможность ручного управления использования памяти, что позволяет сделать код более эффективным. Для создания графического отображения игры на экране пользователя используется библиотека SFML (Simple and Fast Multimedia Library), которая обеспечивает высокую производительность и простоту в использовании.

Язык программирования C++ является мощным и широко используемым языком, который обладает рядом особенностей, положительных и отрицательных сторон.

Основные особенности языка программирования C++:

Мультипарадигменность: C++ поддерживает несколько парадигм программирования, включая процедурное, объектно-ориентированное и обобщенное программирование. Это позволяет разработчикам выбирать подход, наиболее подходящий для конкретной задачи.

Эффективность: C++ предоставляет близкое к машинному коду управление памятью и низкоуровневые операции, позволяющие разработчикам создавать быстрые и эффективные программы. Он предоставляет возможность непосредственного доступа к памяти и оптимизации, что особенно важно для систем с ограниченными ресурсами или высоких требованиях к производительности.

Объектно-ориентированное программирование: C++ поддерживает концепции объектно-ориентированного программирования, такие как инкапсуляция, наследование и полиморфизм. Это позволяет создавать модульные и повторно используемые кодовые блоки, улучшает структурирование программ и облегчает сопровождение.

Богатая стандартная библиотека: C++ предлагает обширную стандартную библиотеку, включающую контейнеры данных, алгоритмы, потоки ввода-вывода, работу с файлами и другие функциональные возможности. Это позволяет разработчикам использовать готовые инструменты и повышает производительность разработки.

Переносимость: C++ является кросс-платформенным языком программирования, что означает, что программы, написанные на C++, могут быть скомпилированы и запущены на различных операционных системах и аппаратных платформах.

Большое количество положительных сторон данного языка программирования также влечет за собой некоторые недостатки, такие как

Сложность: C++ является достаточно сложным языком программирования, требующим от разработчика глубокого понимания его особенностей и возможностей. Это может создавать более крутой кривую обучения и затруднять начинающим программистам освоение языка.

Уязвимость к ошибкам: из-за своей гибкости и возможности непосредственного доступа к памяти, C++ может быть более подвержен ошибкам и уязвимостям в коде, таким как неконтролируемое переполнение буфера или утечки памяти. Неправильное использование указателей и ссылок может привести к ошибкам времени выполнения и нестабильной работе программы.

Большой объем кода: использование C++ может потребовать большего объема кода по сравнению с некоторыми другими языками программирования. Некоторые операции, такие как управление памятью, требуют дополнительных строк кода, что может привести к более длинному и сложному коду.

Отсутствие автоматической сборки мусора: C++ не предоставляет встроенного механизма автоматической сборки мусора, что означает, что разработчику необходимо самостоятельно управлять выделением и освобождением памяти. Это требует от программиста более внимательного контроля и может приводить к ошибкам, связанным с утечками памяти или некорректным освобождением ресурсов.

Большой набор возможностей и функций: в C++ существует множество функций и возможностей, что может привести к перегрузке и сложности выбора наиболее подходящего подхода или функционала. Это может затруднить разработку и понимание кода, особенно для менее опытных разработчиков.

В качестве интегрируемой среды разработки используется Microsoft Visual Studio 2022. Microsoft Visual Studio (VS) - это интегрированная среда разработки (Integrated Development Environment, IDE), предоставляемая компанией Microsoft. Она предназначена для разработки различных типов приложений, включая приложения для Windows, веб-приложения, мобильные приложения, игры и другие.

Плюсы Visual Studio:

Обширный функционал: Visual Studio обладает множеством функций и инструментов, которые позволяют разработчикам создавать и отлаживать приложения различных типов и для разных платформ. Он предлагает интегрированную отладку, автоматическое завершение кода, инструменты для анализа кода, поддержку версий, дизайнеры пользовательского интерфейса и многое другое.

Широкая поддержка языков программирования: Visual Studio поддерживает множество языков программирования, включая C++, C#, Visual Basic, F#, JavaScript, Python и многие другие. Это дает разработчикам возможность выбирать наиболее подходящий язык для своего проекта.

Интеграция с экосистемой Microsoft: Visual Studio хорошо интегрирован с другими продуктами и сервисами Microsoft, такими как Azure, SQL Server, Team Foundation Server и Microsoft 365. Это облегчает разработку и развертывание приложений на платформе Microsoft.

Расширяемость: VS предлагает возможность расширения через дополнения, которые позволяют разработчикам добавлять дополнительные функциональные возможности и инструменты в среду разработки. Существует обширный каталог расширений, которые могут помочь улучшить процесс разработки и повысить производительность.

Минусы Visual Studio:

Ресурсоемкость: Visual Studio является мощным и комплексным инструментом, который требует значительных ресурсов компьютера для его работы. Это может привести к замедлению работы на медленных компьютерах или тех, где доступны ограниченные ресурсы.

Сложность и крутая кривая обучения: из-за своего богатого функционала, Visual Studio может быть сложным для начинающих программистов. Он требует времени и усилий, чтобы полностью овладеть всеми его возможностями и использовать их эффективно.

Платность: в зависимости от используемой версии, Visual Studio может быть платным продуктом. Более расширенные и профессиональные версии IDE могут иметь высокую стоимость, особенно для некоммерческих или небольших проектов. Однако Microsoft также предлагает бесплатные версии, такие как Visual Studio Community, которые предоставляют широкий функционал для индивидуальных разработчиков и малых команд.

Ограниченность платформой: Visual Studio является продуктом, ориентированным на разработку приложений для платформы Microsoft. В результате, если вам требуется создание приложений для других платформ, таких как macOS или Linux, вам может потребоваться использовать другие инструменты разработки.

Размер установочных файлов: полноценная установка Visual Studio может занимать значительное пространство на жестком диске, особенно учитывая многочисленные компоненты и пакеты, включенные в IDE. Это может быть проблематично для устройств с ограниченным местом на диске.

Также для создания графического интерфейса была выбрана библиотека SFML. SFML (Simple and Fast Multimedia Library) - это кросс-платформенная библиотека для разработки мультимедийных и графических приложений. Она предоставляет простой и удобный интерфейс для работы с графикой, звуком, сетью и вводом пользователей. Вот некоторые из плюсов и минусов использования SFML:

Плюсы SFML:

Простота использования: SFML разработана с учетом простоты и понятности интерфейса. Она имеет простую и интуитивно понятную структуру классов и функций, что упрощает разработку приложений.

Кросс-платформенность: SFML является кросс-платформенной библиотекой, поддерживающей Windows, macOS, Linux и другие платформы. Это позволяет создавать приложения, которые могут работать на разных операционных системах без необходимости внесения больших изменений.

Хорошая производительность: SFML предоставляет аппаратное ускорение графики и звука, что позволяет достичь высокой производительности визуализации и обработки звука.

Поддержка различных форматов: SFML поддерживает широкий спектр форматов изображений, звуковых файлов и шрифтов. Это позволяет использовать разнообразные ресурсы для создания приложений.

Поддержка множества языков программирования: SFML имеет официальные привязки к различным языкам программирования, включая C++, C#, Python, Ruby и другие. Это позволяет разработчикам выбирать наиболее удобный язык для создания приложений.

Минусы SFML:

Ограниченные возможности: SFML является относительно простой библиотекой, и в ней может не хватать некоторых продвинутых функций, которые могут быть доступны в других библиотеках или фреймворках.

Отсутствие активной поддержки: SFML обновляется и поддерживается сообществом разработчиков, но у нее нет такой широкой поддержки и активности, как у некоторых других библиотек.

Отсутствие графического редактора: SFML не предоставляет инструментов для создания графических ресурсов или редактирования изображений. Для этого вам потребуется использовать сторонние программы или ресурсы.

Ограниченная сетевая функциональность: SFML предоставляет базовые сетевые возможности, но может оказаться недостаточно для разработки сложных сетевых приложений.

Несмотря на некоторые ограничения, SFML является популярным выбором для разработчиков игр и графических приложений, особенно для тех, кто предпочитает простоту использования и хорошую производительность.

# Программная реализация

* 1. **Физическая структура**

Физическая структура программного обеспечения относится к организации и размещению компонентов программы на физическом уровне. Это включает в себя расположение файлов и каталогов, связанных с программой, на диске, а также организацию модулей, компонентов и зависимостей между ними. Физическая структура определяет, как программное обеспечение физически представлено и хранится на компьютере или в распределенной среде, включая файловую систему, аппаратное обеспечение и развертывание компонентов по различным узлам сети. Она включает в себя следующие аспекты:

Файловая структура: физическая структура программного обеспечения определяет, как файлы и каталоги, связанные с программой, организованы на диске. Это может включать исходный код программы, исполняемые файлы, библиотеки, ресурсы (такие как изображения, звуки и т. д.), конфигурационные файлы и другие ресурсы, необходимые для работы программы.

Модули и компоненты: программное обеспечение может быть организовано в виде различных модулей или компонентов, которые выполняют определенные функции или задачи. Физическая структура определяет, как эти модули и компоненты организованы на диске и как они взаимодействуют друг с другом.

Зависимости и связи: физическая структура программного обеспечения также определяет зависимости и связи между различными компонентами. Например, программный модуль может зависеть от других модулей или библиотек, и эти зависимости должны быть правильно управляемыми и удовлетворены для корректной работы программы.

Распределение и развертывание: в случае распределенных систем или клиент-серверных приложений физическая структура определяет, как компоненты программного обеспечения распределены по разным узлам сети. Она также включает в себя инфраструктуру и ресурсы, необходимые для развертывания и запуска программы.

Аппаратное обеспечение: физическая структура программного обеспечения может зависеть от аппаратного обеспечения, на котором оно будет запускаться. Например, различные компоненты программы могут быть оптимизированы для работы на конкретных типах процессоров или операционных систем.

Корректная и эффективная организация физической структуры программного обеспечения играет важную роль в обеспечении его производительности, масштабируемости, управляемости и обслуживаемости.

Файл Menu.cpp

int main(); - функция, в которой реализован основной цикл рендеринга игрового окна.

Text drawText(RenderWindow& window, String text, Vector2f position, int number, bool isMouseOver); - функция, использующаяся для отображения текста на экране

void readResults(int& LAST\_RESULT, int& BEST\_RESULT); - функция, которая читает данные из файла, хранщего наилучший и последний результат игры.

int readVolumeFromFile(); - функция, которая читает из файла установленное значение громкости.

Vector2i getMousePosition(RenderWindow& window, Vector2i position); - функция, возвращающая относительное положение мыши на экране.

int convertFromPositionToVolume(Vector2i position); - функция, преобразующая целочисленное значение громкости в координаты.

Vector2f convertFromVolumeToPosition(int volume, Object sliderLine); - функция, преобразующая координаты ползунка громкости в целочисленное значение.

void writeVolumeToFile(int volume); - функция, записывающая целочисленное значение громкости с текстовый документ.

void drawBar(RenderWindow& window, vector<int> num, int state); - функция, которая отрисовывает ползунок громкости на экране.

void drawPlayGround(RenderWindow& window, int state); - функция, которая отрисовывает игровое поле на экране.

void showCurrentResult(RenderWindow& window, int CURRENT\_RESULT, int BEST\_RESULT); - функция, выводящая текущий результат на экран монитора.

void writeToFileResults(int LAST\_RESULT); - функция, записывающая последний результат в текстовый файл.

bool isMouseOver(RenderWindow& window, Text text); - функция, проверяющая находится ли курсор мыши поверх текста.

Файл Move.h

bool MoveRight(vector<vector<vector<int>>>& nums, int & CURRENT\_RESULT); - функция, перемещающая плитки на игровом поле в правую сторону.

bool MoveLeft(vector<vector<vector<int>>>& nums, int & CURRENT\_RESULT); - функция, перемещающая плитки на игровом поле в левую сторону.

bool MoveDown(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT); - функция, перемещающая плитки на игровом поле вниз.

bool MoveUp(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT); - функция, перемещающая плитки на игровом поле вверх.

bool checkGameOver(vector<vector<vector<int>>> nums); - функция, проверяющая состояние игрового поле на поражение пользователя.

bool checkIsWinner(vector<vector<vector<int>>>& nums, int state); - функция, проверяющая состояние игрового поле на победу пользователя.

Файл Nums.h

vector<vector<vector<int>>> genNums(int level); - функция, генерирующая пустое игровое поле.

void genFnum(vector<vector<vector<int>>>& nums); - функция, вставляющая первоначальное значение игровых плиток при создании игрового поля.

void newRandNum(vector<vector<vector<int>>>& nums); - функция, генерирующая случайное значение плитки на игровом поле.

Подробная информация о файлах и функциях располагается в приложении А.

# 3.2. Описание разработанных модулей

Компьютерная игра «Слияние чисел» представляет собой комплекс разработанных модулей. Модули взаимодействуют между собой.

Проект включает в себя:

Специальный модуль для импортирования

библиотек и создания пространства имен «includes.h» - заголовочный файл, импортирующий сторонние библиотеки.

Класс игрового поля «Playground.h» - хранит в себе информацию о состояние игрового поля.

Класс «Object.h» - реализует любые объекты, использующие текстуры.

Класс «Button.h» - представляет собой структуру кнопок.

Полный листинг программы находятся в приложении А.

# 4.Тестирование

Тестирование программного обеспечения (ПО) - это процесс проверки программного продукта с целью выявления дефектов, ошибок и недостатков, а также оценки его качества и соответствия требованиям. Оно выполняется с помощью специальных тестовых сценариев, которые создаются для проверки функциональности, производительности, безопасности и других аспектов программы.

Тестирование ПО включает в себя несколько ключевых этапов:

Планирование тестирования: В этом этапе определяются цели и задачи тестирования, составляется тестовая стратегия и план, а также выбираются методы и инструменты тестирования.

Разработка тестовых случаев: На основе требований к программному продукту создаются тестовые случаи, которые описывают конкретные шаги и ожидаемые результаты тестирования. Это включает функциональные, нефункциональные, регрессионные, а также другие типы тестов.

Выполнение тестов: Тестовые случаи выполняются с помощью специального тестового окружения или инструментов. Результаты тестирования фиксируются, и любые обнаруженные дефекты регистрируются в системе отслеживания ошибок.

Анализ результатов: Результаты тестирования анализируются, и производится оценка соответствия программного продукта требованиям и ожиданиям. Выявленные дефекты рассматриваются и исправляются разработчиками.

Повторное тестирование и регрессионное тестирование: После исправления дефектов выполняется повторное тестирование для проверки исправлений. Кроме того, регрессионное тестирование выполняется для убеждения в том, что внесенные изменения не вызвали новых ошибок или не повлияли на работу других функций.

Программное тестирование имеет несколько целей:

Гарантировать, что программное обеспечение соответствует требованиям и спецификациям.

Проверить, что программное обеспечение работает правильно и ожидаемым образом.

Обнаружить и исправить ошибки и дефекты в программном обеспечении.

Улучшить качество программного обеспечения и повысить уровень удовлетворенности пользователей.

Тестирование ПО является важной частью процесса разработки и обеспечивает достоверность и надежность программного продукта перед его выпуском или внедрением в эксплуатацию.

При тестировании программного обеспечения могут возникать различные типы ошибок. Некоторые из наиболее распространенных ошибок включают:

Функциональные ошибки: Это ошибки, при которых программное обеспечение не выполняет требуемую функциональность или выполняет ее неправильно. Например, кнопка не реагирует на нажатие или программа выдает неверные результаты.

Ошибки ввода/вывода: Эти ошибки связаны с некорректной обработкой ввода или вывода данных. Например, программа может неправильно обрабатывать ввод пользователя или некорректно форматировать выводимую информацию.

Ошибки производительности: Эти ошибки возникают, когда программа работает медленно или недостаточно эффективно. Например, загрузка страницы веб-приложения может занимать слишком много времени, или игра может иметь низкую частоту кадров.

Ошибки безопасности: Эти ошибки связаны с уязвимостями программного обеспечения, которые могут быть использованы злоумышленниками для несанкционированного доступа или атаки на систему. Например, недостаточная проверка прав доступа или неправильная обработка входных данных может привести к уязвимостям.

Ошибки совместимости: Эти ошибки возникают, когда программное обеспечение несовместимо с определенными операционными системами, конфигурациями или другими программами. Например, программа может работать некорректно на определенной версии операционной системы или взаимодействовать неправильно с другими приложениями.

Тестирование программы наглядно показывает таблица 2.

Таблица 2 – Тестирование программного средства

|  |  |  |
| --- | --- | --- |
| Тестируемая операция | Требуемый результат | Полученный результат |
| Запуск программы | Появление окна меню  со всеми ее компонентами, загрузка текстур программы. | Через несколько  секунд после открытия приложения появилось окно  меню. |
| Выбор кнопки «Играть» | Появление окна выбора игрового режима | Открытие окна с кнопками для выбора определенного режима игры. |
| Выбор кнопки  «Настройки» | Открытие окна настроек с ползунком изменения громкости музыки | Открылось окно настроек, отобразился ползунок для настройки громкости звука. |
| Выбор кнопки «Рестарт» | Очистка игрового поля, создание нового вектора для игровых плиток | Отобразилось новое игровое поле, текущий результат стал равен 0, наилучший результат обновился. |
| Выбор кнопки  «Домик» | Появление окна меню  со всеми ее компонентами, загрузка текстур программы. | Отобразилось окно меню. |
| Выбор кнопки стрелочки | Перемещение плиток в сторону, соответствующую направлению стрелки на кнопке | Плитки на игровом поле переместились в сторону, соответствующую направлению стрелки на нажатой кнопке. |
| Нажатие клавиш WASD на клавиатуре | Перемещение плиток на игровом поле в определенную сторону, в зависимости от нажатой клавиши. | Плитки на игровом поле были перемещены в необходимую сторону. |
| Передвижение ползунка | Громкость музыки пропорционально изменяется в соответствии с положением ползунка на шкале. | Громкость музыки была изменена в соответствии с положение ползунка. |
| Нажатие на кнопку выбора режима | В соответствии с выбранным режимом отображено окно с игровым полем определенного размера и другими компонентами игрового окна. | Появилось окно с игровым полем с размером, соответствующим выбранному режиму, отобразились остальные компоненты игрового поля. |

В результате тестирования программы ошибок, которые могли бы привести к некорректной работе, обнаружено не было.

# Применение

* 1. **Руководство пользователя**

Руководство пользователя (User Manual) - это документ, предназначенный для ознакомления пользователей с программным или аппаратным продуктом и предоставления им инструкций по его использованию. Оно содержит информацию о функциях, настройках, операциях и других аспектах продукта, которая помогает пользователям получить максимальную пользу от него. Руководство пользователя обычно включает следующие элементы: введение, установка и настройка, основные функции и операции, расширенные настройки, решение проблем, безопасность и предупреждения.

Руководство пользователя разрабатывается с учетом целевой аудитории и должно быть написано простым и понятным языком, чтобы пользователи могли легко ориентироваться и использовать продукт без затруднений.

Для запуска игры пользователю необходимо открыть соответствующую папку, найти в ней файл с иконкой игры и запустить его. После запуска файла можно приступать к игре. После запуска программы появится главное меню (рисунок 5).

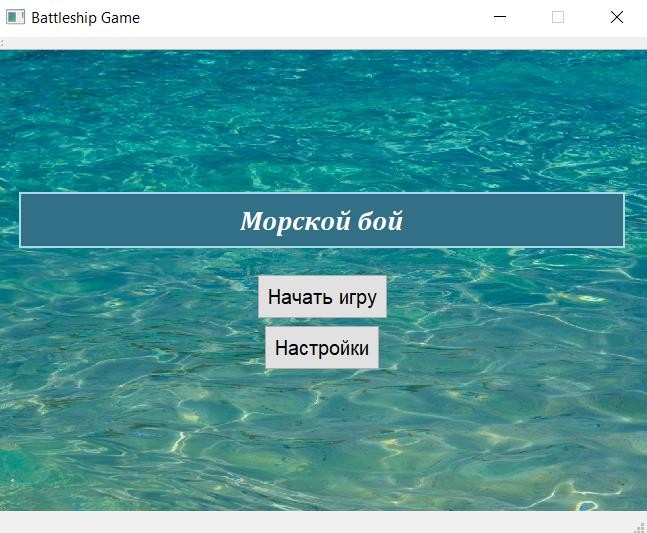


Рисунок 5 – Главное меню программы

В главном меню пользователь может начать игру или нажать клавишу настройки.

При выборе «Настройки», появляется окно, в котором пользователь может изменять значение громкости музыки во время игрового процесса. После всех изменений вы сможете сохранить настройки и вернуться в главное меню (рисунок 6)

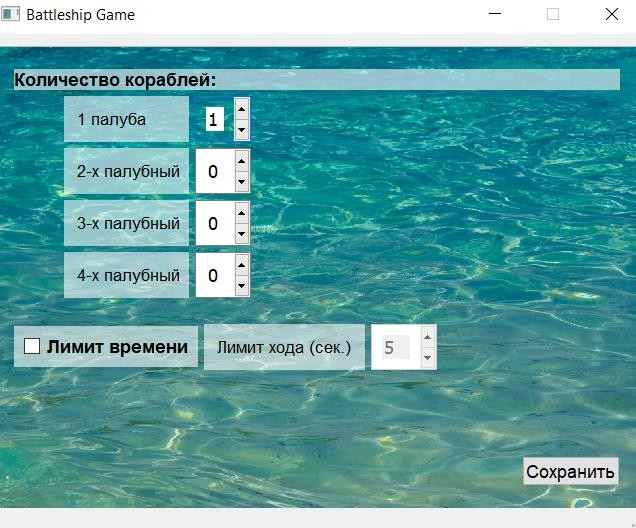


Рисунок 6 – Настройки игры

Выбирая пункт «Играть», пользователю предоставляется окно с выбором режима игры, каждый режим отвечает за размер игрового поля (рисунок 7).



Рисунок 7 – Выбор игрового режима

После выбора необходимого режима перед пользователем будет показано игровое окно, где отображено игровое поле, текущий и наилучший результаты (рисунок 8).

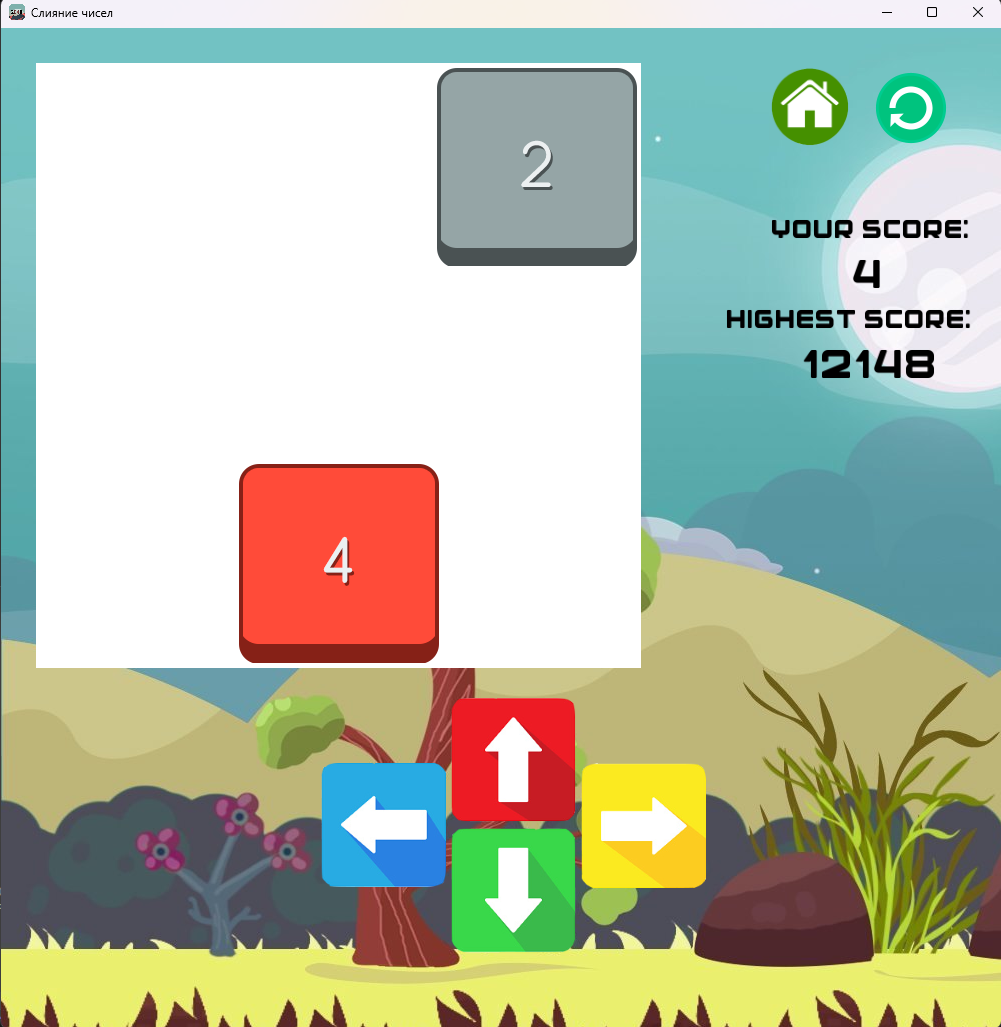


Рисунок 8 – Начало игрового процесса

В данном окне пользователь может управлять плитками на игровом поле путем нажатия клавиш WASD на клавиатуре, а также при помощи кнопок на экране (рисунок 9).

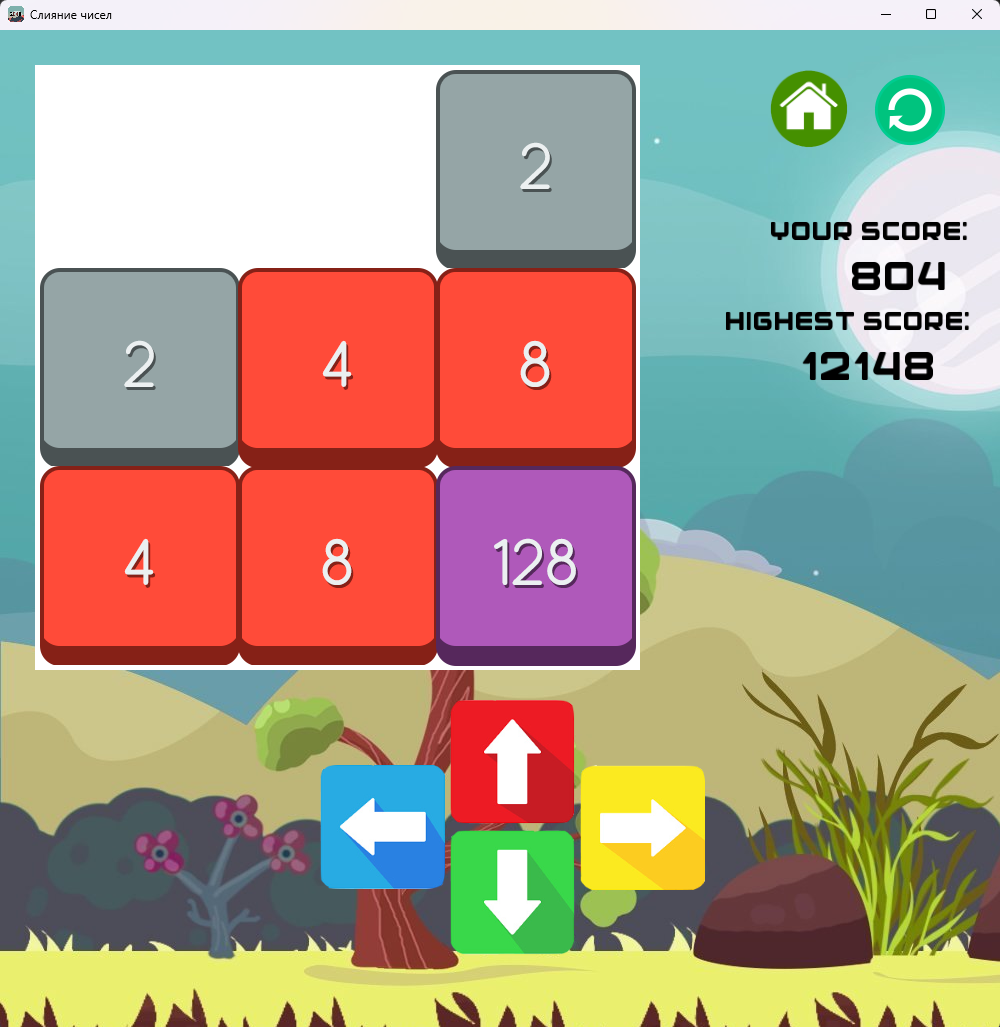


Рисунок 9 – Игровое после нескольких ходов

В случае, если пользователь хочет сменить игровой режим или зайти в настройки, то пользователь может нажать на кнопку «Домик», которая вернет его в главное меню. Также в случае, если пользователь хочет начат игру заново в данном режиме, то он может нажать на кнопку «Рестарт», тогда игра будет перезапущена.

В ходе игрового процесса пользователь игра может быть завершена в двух случаях. В первом случае пользователь набрал максимальное количество очков, предусмотренных для данного режима (2048 – для режима с размером игрового поля 3x3, 4096 – для режимов с размером игрового поля 4x4 и 5x5), тогда игра считается выигранной (рисунок 10).

Второй случай: на игровом поле не осталось возможных действий, тогда игра считается проигранной (рисунок 11).

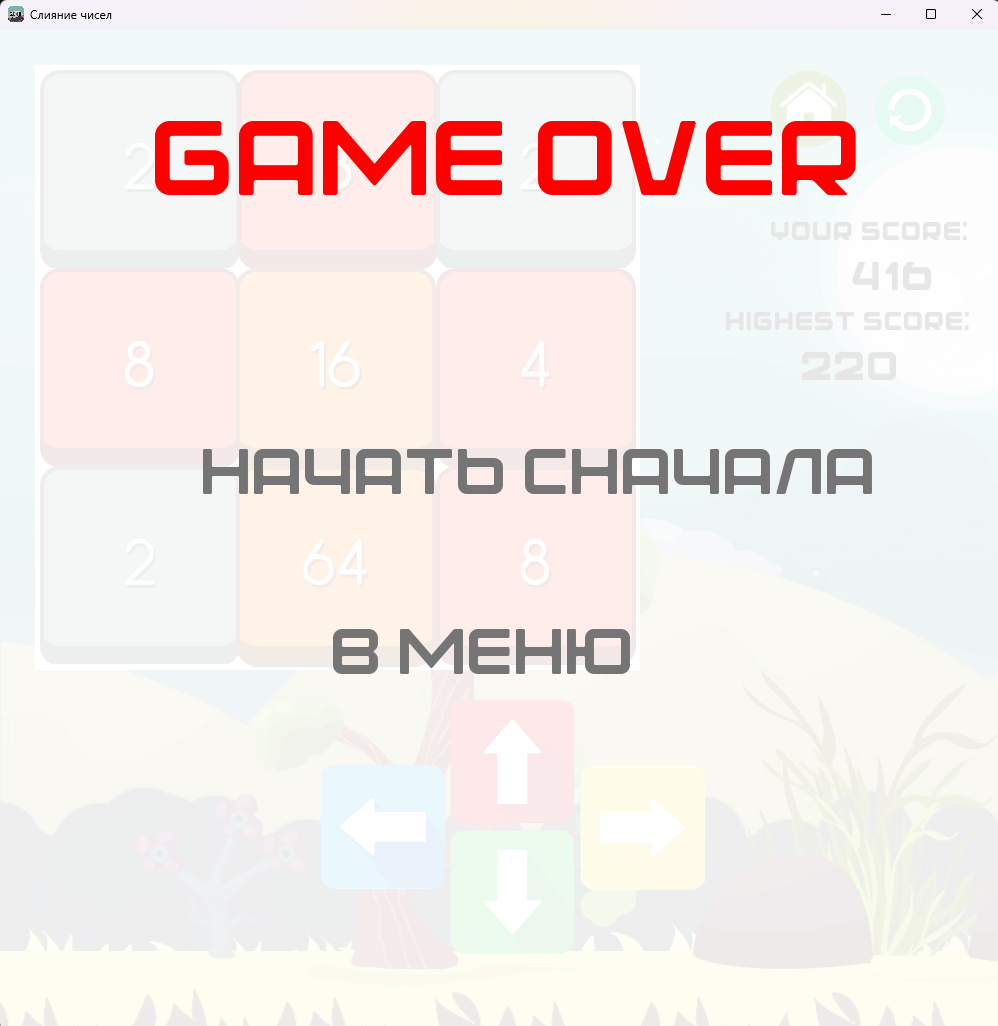


Рисунок 11 – Игровое окно в случае проигрыша

# Заключение

В ходе курсового проекта была достигнута цель проектирования – создание компьютерной игры «Морской бой».

При разработке программы были использованы стандартные компоненты среды разработки, консоль, которые были и использованы при создании средств управления, редактирования или отображения. Курсовой проект реализован максимально удобно как для пользователей, так и для разработчиков несмотря на то, что разработка данной программы оказалась достаточно сложной.

Программному средству необходимы малые требования к устройству. Также с многочисленными подсказками и пояснениями программа может использоваться даже самыми неопытными пользователями, работа с программным средством максимально комфортна, удобна и понятна, к тому же доставляет удовольствие. Программа вполне может сравниться с современными аналогами, так как имеет хороший набор функций.

В результате выполнения курсового проекта были усовершенствованы знания по работе с языком программирования C++. Были получены знания по работе с различными модулями. Данный проект оказался полезным и точно сможет помочь в дальнейшем изучении данной среды программирования. Благодаря реализации данного курсового проекта были приобретены практические и закреплены теоретические навыки и знания, и получено умение самостоятельного анализа поставленной задачи.

Данная пояснительная записка выполнена в соответствии с ГОСТами и требованиями, предъявляемыми к технической документации.

В результате выполнения курсового проекта были приобретены и проверены на практике дополнительные знания о языке программирования C++, приобретены навыки использования научно-технической информации, умение самостоятельно анализировать поставленную задачу и разбивать её на подзадачи, закреплены навыки работы в среде разработки Visual Studio.

Все задачи в процессе разработки курсового проекта были решены. Было спроектировано простое и понятное меню; окно с игровым полем, включающее следующие функции: ввод и обработка цифр, возможность автозаполнения, проверки игрового поля, выбора уровня сложности, записи результата; окно помощи со справочной информацией об игре; форма с показом истории результатов игры.

# Список использованных источников

Литература

1. Довбуш, Галина Visual C++ на примерах / Галина Довбуш, Анатолий Хомоненко. - М.: БХВ-Петербург, 2012. - 528 c.
2. Культин Н.Б. Основы программирования в С++. – СПб.:БХВ – Петербург, 2003. - 608 с.: ил.
3. Оформление курсовых и дипломных проектов. / Сост. И.М.Снежкова. – Мн : МГВРК, 2003.-231 с.

Интернет-ресурсы

1. Компьютерная игра – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Компьютерная\_игра. Дата доступа: 14.05.2022.
2. Аналог – [Электронный ресурс]. – Режим доступа: https://dic.aca- demic.ru/dic.nsf/dmitriev/56/аналог. Дата доступа: 15.05.2022.
3. Информационная база – [Электронный ресурс]. – Режим доступа: https://stu- dopedia.su/9\_8372\_informatsionnaya-baza.html. Дата доступа:

15.05.2022.

1. Алгоритм – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/p ost/591599/comments/. Дата доступа: 12.04.2022.
2. Клуб программистов [Электронный ресурс] – Режим доступа: http:// [www.programmersclub.ru/lab/.](http://www.programmersclub.ru/lab/) – Дата доступа: 01.05.2021.
3. Форум программистов и сисадминов [Электронный ресурс] – Режим доступа: [http://cyberforum.ru/.](http://cyberforum.ru/) – Дата доступа: 01.05.2021.
4. Руководство пользователя – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Руководство\_пользователя. Дата доступа 24.05.2022.
5. Язык C++ – [Электронный ресурс]. – Режим доступа: https:// docs.mi- crosoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. Дата доступа: 25.05.2022.

# ПРИЛОЖЕНИЕ А

**(обязательное) Код программы**

Модуль Menu.cpp

#pragma once

#include"Button.h"

#include"includes.h"

#include"move.h"

#include"Nums.h"

#include<SFML/Audio/Music.hpp>

Text drawText(RenderWindow& window, String text, Vector2f position, int number, bool isMouseOver);

void readResults(int& LAST\_RESULT, int& BEST\_RESULT);

int readVolumeFromFile();

Vector2i getMousePosition(RenderWindow& window, Vector2i position);

int convertFromPositionToVolume(Vector2i position);

Vector2f convertFromVolumeToPosition(int volume, Object sliderLine);

void writeVolumeToFile(int volume);

void drawBar(RenderWindow& window, vector<int> num, int state);

void drawPlayGround(RenderWindow& window, int state);

void showCurrentResult(RenderWindow& window, int CURRENT\_RESULT, int BEST\_RESULT);

void writeToFileResults(int LAST\_RESULT);

bool isMouseOver(RenderWindow& window, Text text);

enum windows {

menuWindow, settingsWindow, gameWindow, menuOption

};

int main() {

RenderWindow window(VideoMode(1000, 1000), L"Слияние чисел", Style::Default, ContextSettings(0, 0, 8));

Music music;

music.openFromFile("src\\sounds\\backsound.ogg");

music.play();

music.setLoop(true);

int volume = readVolumeFromFile();

music.setVolume(volume);

int LAST\_RESULT;

int BEST\_RESULT;

readResults(LAST\_RESULT, BEST\_RESULT);

//SETTING ICON

Image icon = \*new Image();

icon.loadFromFile("src\\logo\\logo2.png");

window.setIcon(300, 300, icon.getPixelsPtr());

Object bg = \*new Object("src\\bg\\bg2.jpg", \*new Vector2f(0, 0));

Font font;

font.loadFromFile("ofont.ru\_Bowler.ttf");

bool isMenu = true;

bool isSettings = false;

bool showOption = false;

bool isGame = false;

int gameState = 0;

//menu attributes

//Button buttonPlay = \*new Button(\*new Vector2f(300, 500));

//Button buttonSettings = \*new Button(\*new Vector2f(300, 665));

//Button buttonAboutUs = \*new Button(\*new Vector2f(300, 820));

//transition attributes

RectangleShape transitionRect;

transitionRect.setPosition(0, 0);

transitionRect.setSize(\*new Vector2f(1000, 1000));

transitionRect.setFillColor(Color::Black);

int alpha = 0;

Clock clock;

float transitionStart = 0;

bool transition = false;

int windowType;

//settings attributes

Object sliderLine = \*new Object("src\\sliders\\line.png", \*new Vector2f(150, 200));

Object slider = \*new Object("src\\sliders\\slider.png", convertFromVolumeToPosition(volume, sliderLine));

//game attributes

int state;

bool buttonPressed = false;

bool keyPressed = false;

double tempTime;

int CURRENT\_RESULT = 0;

int temp;

readResults(temp, BEST\_RESULT);

bool isGameOver = false;

bool isWinner = false;

bool fstart = true;

vector<vector<vector<int>>> nums;

while (window.isOpen()) {

Event ev;

while (window.pollEvent(ev)) {

if (ev.type == Event::Closed) {

writeToFileResults(CURRENT\_RESULT);

window.close();

}

if (ev.type == Event::MouseButtonReleased) {

if (ev.key.code == Mouse::Left) {

buttonPressed = false;

}

}

}

window.clear(Color(255, 255, 255));

window.draw(bg.sprite);

if (isMenu) {

readResults(LAST\_RESULT, BEST\_RESULT);

Text playText(L"ИГРАТЬ", font, 72);

playText.setFillColor(Color::White);

playText.setPosition(340, 500);

Text settingsText(L"НАСТРОЙКИ", font, 72);

settingsText.setFillColor(Color::White);

settingsText.setPosition(250, 665);

Text aboutText(L"О НАС", font, 72);

aboutText.setFillColor(Color::White);

aboutText.setPosition(350, 820);

Text title(L"CЛИЯНИЕ ЧИСЕЛ", font, 62);

title.setFillColor(Color::Black);

title.setPosition(\*new Vector2f(100, 100));

Object logo("src\\logo\\logo1.png", \*new Vector2f(150, 150));

if (Mouse::isButtonPressed(Mouse::Left) && !buttonPressed) {

if (isMouseOver(window, playText) && !showOption) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = menuOption;

}

else if (isMouseOver(window, settingsText) && !showOption) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = settingsWindow;

}

else if (!transition) {

showOption = false;

}

buttonPressed = true;

}

if (!showOption) {

if (isMouseOver(window, playText)) {

Vector2f pos = playText.getPosition();

playText.setFillColor(Color(117, 117, 117));

playText.setScale(\*new Vector2f(1.05, 1.05));

playText.setPosition(pos.x - 5, pos.y - 5);

}

if (isMouseOver(window, settingsText)) {

Vector2f pos = settingsText.getPosition();

settingsText.setFillColor(Color(117, 117, 117));

settingsText.setScale(\*new Vector2f(1.05, 1.05));

settingsText.setPosition(pos.x - 5, pos.y - 5);

}

if (isMouseOver(window, aboutText)) {

Vector2f pos = aboutText.getPosition();

aboutText.setFillColor(Color(117, 117, 117));

aboutText.setScale(\*new Vector2f(1.05, 1.05));

aboutText.setPosition(pos.x - 5, pos.y - 5);

}

}

window.draw(playText);

window.draw(settingsText);

window.draw(aboutText);

drawText(window, to\_string(LAST\_RESULT), \*new Vector2f(100, 100), 0, false);

drawText(window, to\_string(BEST\_RESULT), \*new Vector2f(200, 100), 0, false);

//window.draw(title);

window.draw(logo.sprite);

}

if (isSettings) {

Object buttonExit = \*new Object("src\\buttons\\home.png", \*new Vector2f(770, 40));

drawText(window, "SETTINGS", \*new Vector2f(400, 50), 0, false);

window.draw(sliderLine.sprite);

if (buttonExit.isMouseOver(window, 0)) {

buttonExit.sprite.setScale(1.05f, 1.05f);

buttonExit.sprite.setPosition(buttonExit.sprite.getPosition().x - 2.f, buttonExit.sprite.getPosition().y - 2.f);

}

window.draw(buttonExit.sprite);

if (Mouse::isButtonPressed(Mouse::Left)) {

Vector2i mouseCoord = getMousePosition(window, Mouse::getPosition());

if (sliderLine.isMouseOver(window, 1)) {

slider.sprite.setPosition(mouseCoord.x-50, sliderLine.sprite.getPosition().y - 18);

volume = convertFromPositionToVolume(mouseCoord);

music.setVolume(volume);

}

else if (buttonExit.isMouseOver(window, 0)) {

if (!transition) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = menuWindow;

}

}

}

window.draw(buttonExit.sprite);

window.draw(slider.sprite);

writeVolumeToFile(volume);

}

if (showOption){

RectangleShape rect;

rect.setSize(\*new Vector2f(1000, 1000));

rect.setFillColor(\*new Color(255, 255, 255, 230));

Font font;

font.loadFromFile("ofont.ru\_Bowler.ttf");

Text gameMode(L"ВЫБЕРИТЕ РЕЖИМ", font, 62);

gameMode.setFillColor(Color::Green);

gameMode.setPosition(\*new Vector2f(180, 150));

Object x3("src\\buttons\\3x3.png", \*new Vector2f(350, 300));

Object x4("src\\buttons\\4x4.png", \*new Vector2f(350, 520));

Object x5("src\\buttons\\5x5.png", \*new Vector2f(350, 740));

if (x3.isMouseOver(window, 0)) {

Vector2f pos = x3.sprite.getPosition();

x3.sprite.setScale(1.05, 1.05);

x3.sprite.setPosition(pos.x - 5, pos.y - 5);

}

else if (x4.isMouseOver(window, 0)) {

Vector2f pos = x4.sprite.getPosition();

x4.sprite.setScale(1.05, 1.05);

x4.sprite.setPosition(pos.x - 5, pos.y - 5);

}

else if (x5.isMouseOver(window, 0)) {

Vector2f pos = x5.sprite.getPosition();

x5.sprite.setScale(1.05, 1.05);

x5.sprite.setPosition(pos.x - 5, pos.y - 5);

}

if (Mouse::isButtonPressed(Mouse::Left)) {

if (x3.isMouseOver(window, 0)) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = gameWindow; state = 3;

}

else if (x4.isMouseOver(window, 0)) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = gameWindow; state = 4;

}

else if (x5.isMouseOver(window, 0)) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = gameWindow; state = 5;

}

}

window.draw(rect);

window.draw(x3.sprite);

window.draw(x4.sprite);

window.draw(x5.sprite);

window.draw(gameMode);

}

if (isGame) {

Object buttonRestart = \*new Object("src\\buttons\\restart.png", \*new Vector2f(870, 40));

Object buttonExit = \*new Object("src\\buttons\\home.png", \*new Vector2f(770, 40));

Object upArrow = \*new Object("src\\buttons\\upArrow.png", \*new Vector2f(450, 670));

Object downArrow = \*new Object("src\\buttons\\downArrow1.png", \*new Vector2f(450, 800));

Object leftArrow = \*new Object("src\\buttons\\leftArrow1.png", \*new Vector2f(320, 735));

Object rightArrow = \*new Object("src\\buttons\\rightArrow1.png", \*new Vector2f(580, 735));

if (!isWinner && !isGameOver)

if (buttonExit.isMouseOver(window, 0)) {

buttonExit.sprite.setScale(1.05f, 1.05f);

buttonExit.sprite.setPosition(buttonExit.sprite.getPosition().x - 2.f, buttonExit.sprite.getPosition().y - 2.f);

}

else if (buttonRestart.isMouseOver(window, 0)) {

buttonRestart.sprite.setScale(1.05f, 1.05f);

buttonRestart.sprite.setPosition(buttonRestart.sprite.getPosition().x - 2.f, buttonRestart.sprite.getPosition().y - 2.f);

}

else if (downArrow.isMouseOver(window, 0)) {

downArrow.sprite.setScale(1.05f, 1.05f);

downArrow.sprite.setPosition(downArrow.sprite.getPosition().x - 2.f, downArrow.sprite.getPosition().y - 2.f);

}

else if (upArrow.isMouseOver(window, 0)) {

upArrow.sprite.setScale(1.05f, 1.05f);

upArrow.sprite.setPosition(upArrow.sprite.getPosition().x - 2.f, upArrow.sprite.getPosition().y - 2.f);

}

else if (leftArrow.isMouseOver(window, 0)) {

leftArrow.sprite.setScale(1.05f, 1.05f);

leftArrow.sprite.setPosition(leftArrow.sprite.getPosition().x - 2.f, leftArrow.sprite.getPosition().y - 2.f);

}

else if (rightArrow.isMouseOver(window, 0)) {

rightArrow.sprite.setScale(1.05f, 1.05f);

rightArrow.sprite.setPosition(rightArrow.sprite.getPosition().x - 2.f, rightArrow.sprite.getPosition().y - 2.f);

}

window.draw(bg.sprite);

window.draw(buttonRestart.sprite);

window.draw(buttonExit.sprite);

window.draw(downArrow.sprite);

window.draw(upArrow.sprite);

window.draw(leftArrow.sprite);

window.draw(rightArrow.sprite);

drawPlayGround(window, state);

for (int i = 0; i < size(nums); ++i) {

for (int j = 0; j < size(nums[i]); ++j) {

if (nums[i][j][0] != 0) {

drawBar(window, nums[i][j], state);

}

}

}

isGameOver = checkGameOver(nums);

isWinner = checkIsWinner(nums, state);

if (ev.type == Event::MouseButtonReleased && ev.mouseButton.button == Mouse::Left) {

buttonPressed = false;

}

if (ev.type == Event::KeyReleased && (ev.key.code == Keyboard::W || ev.key.code == Keyboard::S || ev.key.code == Keyboard::D || ev.key.code == Keyboard::A)) {

keyPressed = false;

}

if (!keyPressed && !isGameOver) {

if (Keyboard::isKeyPressed(Keyboard::D)) {

keyPressed = true;

if (MoveRight(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (Keyboard::isKeyPressed(Keyboard::A)) {

keyPressed = true;

if (MoveLeft(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (Keyboard::isKeyPressed(Keyboard::S)) {

keyPressed = true;

if (MoveDown(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (Keyboard::isKeyPressed(Keyboard::W)) {

keyPressed = true;

if (MoveUp(nums, CURRENT\_RESULT))

newRandNum(nums);

}

}

if (Mouse::isButtonPressed(Mouse::Left) && !buttonPressed && !isGameOver) {

buttonPressed = true;

if (buttonRestart.isMouseOver(window, 0)) {

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = gameWindow;

}

else if (buttonExit.isMouseOver(window, 0)) {

writeToFileResults(CURRENT\_RESULT);

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = menuWindow;

}

else if (upArrow.isMouseOver(window, 0)) {

if (MoveUp(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (leftArrow.isMouseOver(window, 0)) {

if (MoveLeft(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (rightArrow.isMouseOver(window, 0)) {

if (MoveRight(nums, CURRENT\_RESULT))

newRandNum(nums);

}

else if (downArrow.isMouseOver(window, 0)) {

if (MoveDown(nums, CURRENT\_RESULT))

newRandNum(nums);

}

}

showCurrentResult(window, CURRENT\_RESULT, BEST\_RESULT);

if (isGameOver || isWinner) {

// GENERATING TEXT

Object textGameOver("src\\buttons\\over.png", \*new Vector2f(150, 50));

Font font;

font.loadFromFile("ofont.ru\_Bowler.ttf");

RectangleShape rect;

rect.setSize(\*new Vector2f(1000, 1000));

rect.setFillColor(\*new Color(255, 255, 255, 230));

Object restart("src\\buttons\\snachala.png", \*new Vector2f(200, 600));

Object toMenu("src\\buttons\\menu.png", \*new Vector2f(380, 780));

if (restart.isMouseOver(window, 0)) {

Vector2f pos = restart.sprite.getPosition();

restart.sprite.setScale(\*new Vector2f(1.05, 1.05));

restart.sprite.setPosition(\*new Vector2f(pos.x-10, pos.y-10));

}

if (toMenu.isMouseOver(window, 0)) {

Vector2f pos = toMenu.sprite.getPosition();

toMenu.sprite.setScale(\*new Vector2f(1.05, 1.05));

toMenu.sprite.setPosition(\*new Vector2f(pos.x - 5, pos.y - 5));

}

if (Mouse::isButtonPressed(Mouse::Left)) {

if (restart.isMouseOver(window, 0)) {

writeToFileResults(CURRENT\_RESULT);

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = gameWindow;

}

else if (toMenu.isMouseOver(window, 0)) {

writeToFileResults(CURRENT\_RESULT);

transition = true; transitionStart = clock.getElapsedTime().asSeconds(); windowType = menuWindow;

}

}

window.draw(rect);

window.draw(restart.sprite);

window.draw(toMenu.sprite);

window.draw(textGameOver.sprite);

}

}

if (transition) {

int duration = 2;

writeToFileResults(CURRENT\_RESULT);

if (!transition) {

transition = true;

transitionStart = clock.getElapsedTime().asSeconds();

}

else {

float transitionTime = clock.getElapsedTime().asSeconds() - transitionStart;

transitionRect.setFillColor(sf::Color(0, 0, 0, alpha));

if (clock.getElapsedTime().asSeconds() < transitionStart + duration / 2) {

alpha = (transitionTime / (duration / 2)) \* 255;

}

else if (clock.getElapsedTime().asSeconds() < transitionStart + duration) {

if (isMenu && windowType != menuWindow && windowType != menuOption) isMenu = false;

if (isGame && windowType != gameWindow) isGame = false;

if (showOption && windowType != menuOption) showOption = false;

if (isSettings && windowType != settingsWindow) isSettings = false;

switch (windowType)

{

case menuOption: { showOption = true; break; }

case settingsWindow: { isSettings = true;break; }

case menuWindow: { isMenu = true; break; }

case gameWindow: {

isGame = true;

if (fstart) {

nums = genNums(state);

genFnum(nums);

fstart = false;

// nums[2][2][0] = 2048;

}

break; }

default:

break;

}

alpha = 255 - ((transitionTime - (duration / 2)) / (duration / 2)) \* 255;

}

else {

fstart = true;

transition = false;

}

}

window.draw(transitionRect);

}

window.display();

}

}

void drawBar(RenderWindow& window, vector<int> num, int state) {

Image img = Image();

img.loadFromFile("src\\" + to\_string(state) + "\\" + to\_string(num[0]) + ".png");

Texture texture = Texture();

texture.loadFromImage(img);

Sprite sprite = Sprite();

sprite.setTexture(texture);

sprite.setPosition(\*new Vector2f(num[1], num[2]));

window.draw(sprite);

}

void drawPlayGround(RenderWindow& window, int state) {

RectangleShape rect;

rect.setFillColor(Color::White);

rect.setPosition(\*new Vector2f(35, 35));

rect.setSize(\*new Vector2f(605, 605));

window.draw(rect);

}

void showCurrentResult(RenderWindow& window, int CURRENT\_RESULT, int BEST\_RESULT) {

Font font;

font.loadFromFile("ofont.ru\_Bowler.ttf");

Text current\_number(to\_string(CURRENT\_RESULT), font, 40);

current\_number.setFillColor(Color::Black);

current\_number.setStyle(sf::Text::Bold);

current\_number.setPosition(\*new Vector2f(850, 220));

Text text("YOUR SCORE:", font, 25);

text.setFillColor(Color::Black);

text.setStyle(sf::Text::Bold);

text.setPosition(\*new Vector2f(770, 185));

Text text1("HIGHEST SCORE:", font, 25);

text1.setFillColor(Color::Black);

text1.setStyle(sf::Text::Bold);

text1.setPosition(\*new Vector2f(725, 275));

Text highest\_number(to\_string(BEST\_RESULT), font, 40);

highest\_number.setFillColor(Color::Black);

highest\_number.setStyle(sf::Text::Bold);

highest\_number.setPosition(\*new Vector2f(800, 310));

window.draw(current\_number);

window.draw(highest\_number);

window.draw(text);

window.draw(text1);

}

void writeToFileResults(int LAST\_RESULT) {

fstream file;

file.open("result.txt");

int BEST\_RESULT;

file >> BEST\_RESULT;

file >> BEST\_RESULT;

file.close();

if (BEST\_RESULT < LAST\_RESULT)

BEST\_RESULT = LAST\_RESULT;

ofstream fout;

fout.open("result.txt");

fout.clear();

fout << LAST\_RESULT << "\n" << BEST\_RESULT;

fout.close();

}

void writeVolumeToFile(int volume) {

ofstream file;

file.open("properties.txt");

file.clear();

file << volume;

file.close();

}

int readVolumeFromFile() {

int temp;

fstream file;

file.open("properties.txt");

file >> temp;

file.close();

return temp;

}

Vector2i getMousePosition(RenderWindow& window, Vector2i position) {

return position - window.getPosition();

}

int convertFromPositionToVolume(Vector2i position) {

int x = position.x - 150;

return x / 6.5;

}

Vector2f convertFromVolumeToPosition(int volume, Object sliderLine) {

return \*new Vector2f(volume \* 5.8 + 150, sliderLine.sprite.getPosition().y - 18);

}

Text drawText(RenderWindow& window, String text, Vector2f position, int number, bool isOption) {

Font font;

font.loadFromFile("font1.ttf");

Text text1(text, font, 40);

text1.setFillColor(Color(255, 255, 255));

if (isMouseOver(window, text1)) {

text1.setFillColor(Color(176, 176, 176));

}

text1.setStyle(sf::Text::Bold);

switch (number)

{

case 0: {text1.setPosition(position); break; }

case 1: {text1.setPosition(\*new Vector2f(position.x + 115, position.y + 25)); break; }

case 2: {text1.setPosition(\*new Vector2f(position.x + 55, position.y + 25)); break; }

case 3: {text1.setPosition(\*new Vector2f(position.x + 130, position.y + 25)); break; }

default: break;

}

window.draw(text1);

return text1;

}

void readResults(int& LAST\_RESULT, int& BEST\_RESULT) {

fstream file;

file.open("result.txt");

file >> LAST\_RESULT;

file >> BEST\_RESULT;

file.close();

}

bool isMouseOver(RenderWindow& window, Text text)

{

int sizeX = text.getLocalBounds().width+5;

int sizeY = text.getLocalBounds().height+27;

Vector2i mousePos = Mouse::getPosition(window);

if (mousePos.x > text.getPosition().x && mousePos.x <text.getPosition().x + sizeX

&& mousePos.y > text.getPosition().y && mousePos.y < text.getPosition().y + sizeY)

return true;

else

return false;

}

Модуль Button.cpp

#include "Button.h"

Button::Button(Vector2f position)

{

this->img = Image();

this->img.loadFromFile("src\\buttons\\button5.png");

this->texture = Texture();

this->texture.loadFromImage(this->img);

this->sprite = Sprite();

this->sprite.setTexture(this->texture);

this->sprite.setPosition(position);

}

bool Button::isMouseOver(RenderWindow& window)

{

Vector2i mousePos = Mouse::getPosition(window);

if (mousePos.x > this->sprite.getPosition().x && mousePos.x < this->sprite.getPosition().x + 400

&& mousePos.y > this->sprite.getPosition().y && mousePos.y < this->sprite.getPosition().y+150)

return true;

else

return false;

}

Модуль move.cpp

#include "move.h"

#include"includes.h"

#include"Nums.h"

bool MoveRight(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT)

{

int moves = 0;

for (int row = 0; row < nums.size(); ++row){

int i = nums[row].size() - 1;

while (i >= 0) {

if (nums[row][i][0] != 0) {

int j = i + 1;

while (j < nums[row].size() && nums[row][j][0] == 0)

++j;

if (j != i + 1) {

moves++;

}

if (j < nums[row].size() && nums[row][j][0] == nums[row][i][0]) {

// Если найдено одинаковое значение, складываем их

nums[row][j][0] += nums[row][i][0];

CURRENT\_RESULT += nums[row][i][0]\*2;

nums[row][i][0] = 0;

moves++;

}

else {

// В противном случае перемещаем элемент вправо

j--;

nums[row][j][0] = nums[row][i][0];

if (j != i) {

nums[row][i][0] = 0;

}

}

}

i--;

}

}

return moves > 0;

}

bool MoveLeft(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT)

{

int moves = 0;

for (int row = 0; row < nums.size(); ++row) {

int index = 0;

while (index < nums[row].size()) {

if (nums[row][index][0] != 0) {

int j = index-1;

while (j >= 0 && nums[row][j][0] == 0) {

--j;

}

if (j != index - 1) {

moves++;

}

if (j >= 0 && nums[row][index][0] == nums[row][j][0]) {

nums[row][j][0] \*= 2;

CURRENT\_RESULT += nums[row][index][0]\*2;

nums[row][index][0] = 0;

moves++;

}

else {

++j;

nums[row][j][0] = nums[row][index][0];

if (index != j)

nums[row][index][0] = 0;

//moves++;

}

}

++index;

}

}

return moves>0;

}

bool MoveDown(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT)

{

int moves = 0;

for (int col = 0; col < nums.size(); ++col) {

int i = nums[col].size() - 1;

while (i >= 0) {

if (nums[i][col][0] != 0) {

int j = i + 1;

while (j < nums[col].size() && nums[j][col][0] == 0)

++j;

if (j != i + 1) {

moves++;

}

if (j < nums[col].size() && nums[j][col][0] == nums[i][col][0]) {

// Если найдено одинаковое значение, складываем их

nums[j][col][0] += nums[i][col][0];

CURRENT\_RESULT += nums[i][col][0]\*2;

nums[i][col][0] = 0;

moves++;

}

else {

// В противном случае перемещаем элемент вправо

j--;

nums[j][col][0] = nums[i][col][0];

if (j != i) {

nums[i][col][0] = 0;

}

//moves++;

}

}

i--;

}

}

return moves>0;

}

bool MoveUp(vector<vector<vector<int>>>& nums, int& CURRENT\_RESULT)

{

int moves = 0;

for (int col = 0; col < nums.size(); ++col) {

int index = 0;

while (index < nums[col].size()) {

if (nums[index][col][0] != 0) {

int j = index - 1;

while (j >= 0 && nums[j][col][0] == 0) {

--j;

}

if (j != index - 1) {

moves++;

}

if (j >= 0 && nums[index][col][0] == nums[j][col][0]) {

nums[j][col][0] \*= 2;

CURRENT\_RESULT += nums[index][col][0]\*2;

nums[index][col][0] = 0;

moves++;

}

else {

++j;

nums[j][col][0] = nums[index][col][0];

if (index != j)

nums[index][col][0] = 0;

//moves++;

}

}

++index;

}

}

return moves>0;

}

bool checkGameOver(vector<vector<vector<int>>> nums)

{

int a = 0;

bool isEmpty = false;

for (int i = 0; i < nums.size(); ++i) {

for (int j = 0; j < nums[i].size(); ++j) {

if (nums[i][j][0] == 0) {

return false;

}

}

}

return !(MoveDown(nums, a) || MoveUp(nums, a) || MoveLeft(nums, a) || MoveRight(nums, a));

}

bool checkIsWinner(vector<vector<vector<int>>>& nums, int state)

{

if (state == 3 || state == 4)

state = 0;

else if (state == 5)

state = 1;

for (int i = 0; i < nums.size(); ++i) {

for (int j = 0; j < nums[i].size(); ++j) {

if (state == 1 && nums[i][j][0] == 4096 || state == 0 && nums[i][j][0] == 2048) {

return true;

}

}

}

return false;

}

Модуль Nums.cpp

#pragma once

#include"Nums.h"

#include"includes.h"

vector<vector<vector<int>>> genNums(int level)

{

vector<vector<vector<int>>> nums;

int dist = 0;

switch (level) {

case 3: {dist = 198; break; }

case 4: {dist = 148; break; }

case 5: {dist = 118; break; }

default: break;

}

int x = 40; int y = 40;

for (int i = 0; i < level; ++i) {

vector<vector<int>> temp;

for (int j = 0; j < level; ++j) {

vector<int> t;

t.push\_back(0);

t.push\_back(x);

t.push\_back(y);

x += dist;

temp.push\_back(t);

}

x = 40;

y += dist;

nums.push\_back(temp);

if (i == level - 1) {

temp.clear();

}

}

return nums;

}

void genFnum(vector<vector<vector<int>>>& nums)

{

srand(time(NULL));

for (int i = 0; i < 2; ++i) {

const int vals[2]{ 2, 4 };

int x = rand() % (size(nums) - 1);

int y = rand() % (size(nums[0]) - 1);

if (nums[x][y][0] == 0) {

nums[x][y][0] = vals[rand() % 2];

}

else {

--i;

continue;

}

}

}

void newRandNum(vector<vector<vector<int>>>& nums)

{

srand(time(NULL));

int x = rand() % (size(nums) - 1);

int y = rand() & (size(nums) - 1);

int attempts = 0;

bool fl = false;

while (attempts < 100) {

x = rand() % (size(nums) - 1);

y = rand() & (size(nums) - 1);

if (nums[x][y][0] == 0) {

fl = true;

break;

}

++attempts;

}

if (fl)

nums[x][y][0] = 2;

}

Модуль Obejct.cpp

#include"Object.h"

Object::Object(string path, Vector2f position)

{

this->img = Image();

this->img.loadFromFile(path);

this->texture = Texture();

this->texture.loadFromImage(this->img);

this->sprite = Sprite();

this->sprite.setTexture(this->texture);

this->sprite.setPosition(position);

}

Object::Object()

{

}

bool Object::isMouseOver(RenderWindow& window, int type)

{

int width = this->sprite.getLocalBounds().width;

int posX = this->sprite.getPosition().x;

if (type == 1) {

width -= 80;

posX += 40;

}

Vector2i mousePos = Mouse::getPosition(window);

if (mousePos.x > posX && mousePos.x <= posX + width

&& mousePos.y > this->sprite.getPosition().y && mousePos.y < this->sprite.getPosition().y + this->sprite.getLocalBounds().height)

return true;

else

return false;

}

Модуль Playground.cpp

#include "Playground.h"

Playground::Playground(int state)

{

this->img = Image();

this->img.loadFromFile("src\\playgrounds\\" + to\_string(state)+".png");

this->texture = Texture();

this->texture.loadFromImage(this->img);

this->sprite = Sprite();

this->sprite.setTexture(this->texture);

}